Toward a Fully De-identified Biomedical Information Warehouse

Jianhua Liu, PhD; Selnur Erdal, DDS, MS; Scott A. Silvey;
Jing Ding, PhD; John D. Riedel, BS; Clay B. Marsh, MD; Jyoti Kamal, PhD
The Ohio State University Medical Center, Columbus, Ohio

ABSTRACT:
The Information Warehouse at the Ohio State University Medical Center is a comprehensive repository of business, clinical, and research data from various source systems. Data collected here is a valuable resource that facilitates both translational research and personalized healthcare. The use of such data in research is governed by federal privacy regulations with oversight by the Institutional Review Board. In 2006, the Information Warehouse was recognized by the OSU IRB as an "Honest Broker" of clinical data, providing investigators with de-identified or limited datasets under stipulations contained in a signed data use agreement. In order to streamline this process even further, the Information Warehouse is developing a de-identified data warehouse that is suitable for direct user access through a controlled query tool that is aimed to support both research and education activities. In this paper we report our findings on performance evaluation of different de-identification schemes that may be used to ensure regulatory compliance while also facilitating practical database updating and querying. We also discuss how date-shifting in the de-identification process can impact other data elements such as diagnosis and procedure codes and consider a possible solution to those problems.

BACKGROUND

After the adoption of electronic medical record (EMR) systems in an academic medical center in support of clinical operations, data from the EMR becomes a valuable resource for clinical and translational research[1]. This data, combined with other information such as genomic data, lays the foundation for personalized healthcare[2]. The use of patient records in research and education is governed by federal privacy regulations[3] (such as HIPAA and the Common Rule), that constrain the use of identifiable patient information in research activities. Investigators can obtain identifiable data for research use after an Institutional Review Board (IRB) approval is obtained. Obtaining an IRB approval, however, can be a time consuming process that may at times discourage research activities.

Therefore, to support both research and education, it is desirable to have a completely de-identified database (DDB) that is similar in data structure to the identifiable version (IDB). This DDB must be HIPAA compliant in terms of the masking or removal of Protected Health Information (PHI), and should ideally be capable of defending against re-identification attacks. Structural similarity between IDB and DDB is beneficial to ensure that previously developed applications against the IDB will be applicable on the DDB with minimal modification.

In order to support research and education activities using medical records, efforts have been made to de-identify clinical data in both structured[4,5,6,7] and unstructured[4,8,9] data. Methods have been described using a hash function to build a de-identified bio-repository[5]. Other methods have been reported on either de-identification schemes[10] or trying to guarantee anonymity[11]. De-identification of medical free text has been studied and published with great progress in the past years.

In preparation of building a de-identified Information Warehouse[12] (IW) at the Ohio State University Medical Center (OSUMC), we report here our evaluation of the impact of different de-identification methods on database performance and related issues. This resource is intended to provide services to investigators and educators under the OSUMC IRB approved Honest Broker Protocol.

THE OSUMC HONEST BROKER PROTOCOL

The IW's "Honest Broker" status as a provider of de-identified clinical data for research purposes was approved as an annually reviewed procedural protocol by the Ohio State University Biomedical IRB in April 2006. Under this protocol (the Honest Broker Protocol[13], or HBP), a researcher typically signs a data use agreement with the IW when they request a de-identified or limited dataset. IW data analysts prepare the resulting data by removing PHI and deliver the data set as a bundle of coherently linked files to the researcher. In a de-identified dataset, dates are recorded as time intervals from a patient's first visit. Actual dates, zip codes, and ages over 90 can be included in the limited dataset[3].

To safeguard against accidental identification, resulting datasets with less than 25 records are not delivered, only aggregate numbers are reported to the requestor. Re-queries based on previous results are not allowed. De-identified codes are changed between datasets to inhibit longitudinal study of particular subjects. Furthermore, should an inadvertent identification occur, the Data Use Agreement stipulates

that the investigator will immediately seek IRB oversight.

METHOD AND RESULTS

The planned system architecture is depicted in Figure 1. Another instance of IDB is replicated and updated using Change Data Capture for near real-time update (IDB2). IDB2 serves as a backup for IDB as well as a de-identified data feed for DDB. Original linkage tables (mapping between identified and de-identified codes) are maintained in IDB2 only. Only secondary mapping (see method sections below) will be stored in DDB when necessary.

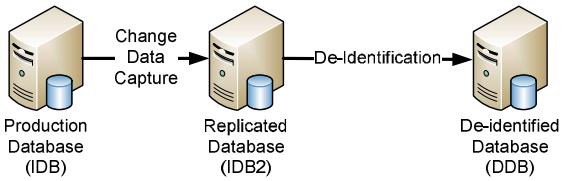

Figure 1: System Architecture for building a de-identified Information Warehouse.

This work is to address two specific issues that are required by the HBP[7]: complete de-identification of data and inhibiting the use of de-identified data for longitudinal study. To prevent longitudinal study (akin to human subject research), we require a viable mechanism to periodically generate an entirely new set of DID's throughout the DDB. This process must perform well enough to complete within an overnight session even as the warehouse continues to grow in size throughout the foreseeable future. Query performance of the DDB is also a consideration as the system must remain as practical as the IDB for performing complex queries and data mining tasks. With these constraints in mind, we consider a variety of possible de-identification mechanisms for the DDB and seek to evaluate the relative performance impacts of each method. We also consider mechanisms for date shifting to preserve inter-event time intervals in the DDB and explore how this can impact some analysis problems.

For de-identification of structured data, we choose to perform tests on identifiers such as Medical Record Number (MRN), encounter number (ENC), and accession number (ACSN) that can be used to uniquely identify a patient. The following de-identification schemes are tested:

1. Secure Hash Algorithm (SH) – identifier (ID) is hashed into a unique string serving as the de-identified identifier (DID) [15]. For testing purpose, SHA-1 algorithm is used to generate the 40-byte long DID. Subsequent database update will not change the DID. Therefore, the DID will keep constant for the life of the DDB.

2. Random Mapping (RM) – actual identifier code (ID) is directly mapped to a random number (10 digits) as the DID. This process produces a DDB that is classified as a "coded" dataset and therefore is subjected to more restricted use restrictions for research purposes[3]. This DID can only be changed by a complete rebuild of the DDB or a re-mapping followed by an update.

3. Hashing-Mapping (HM) – As the combination of 1 and 2 above, the ID is first hashed into a unique string and then each string is assigned a 10 digit random number as the DID. This DID can subsequently be changed by DDB update or rebuild to meet the requirement of inhibiting longitudinal studies.

4. Hashing-Mapping-and-Re-Mapping (HMM) – ID is first hashed into a string, followed by mapping the string to a permanent primary unique random number. This in turn is mapped to a secondary unique random number that is transient between re-coding cycles of the DDB. Both random numbers are 10 digits long. Data in DDB is stored with the primary number and query results are shown with the secondary number serving as the DID through database utilities.

We intend to compare RM and SH for expense on the use of a hashing function and use SH as a standard to evaluate the performance of the HM and HMM methods. Query execution time is gathered using Oracle's SQL Trace Facility and TKPROF utility. CPU time for query execution is collected 4 times, with the first data point discarded and the other 3 recorded and averaged.

All performance tests are performed on a Sun Fire V445 server running Solaris 10 and Oracle Database 11gR1. Secure Hash Algorithm SHA-1 is used in this work. For both SHA-1 and the random number generator, Oracle's DBMS.Crypto toolkit is used. All tables are created with table logging turned off.

To test SH, a database function is created to take an ID as input and then output the hashed string as the DID. Tables containing mappings of MRN-DID and HashedString-1stDID-2ndDID are created for RM and HMM, respectively. HM shares the same table with HMM by using the HashedString-1stDID pair in table.

Table creation is evaluated by using the following SQL statements:

SH:

```
INSERT INTO dest_table
SELECT sha_1(mrn) AS mrn, other_fields
  FROM orig_table;
```

RM:

```
INSERT INTO dest_table
SELECT mapping.did, orig.other_fields
  FROM orig_table, mapping_table
 WHERE orig.mrn=mapping.mrn;
```

HM/HMM

```
INSERT INTO DEST_TABLE
SELECT mapping.did, orig.other_fields
  FROM orig_table, mapping_table
 WHERE sha_1(orig.mrn)=mapping.hash_string;
```

Table 1: Table Creation Time for initial loading of 100K and 1 Million records (seconds)

# Rec.	Method	1	2	3	Average
100K	RM	9.58	9.66	9.53	9.59
	SH	15.58	16.02	16.11	15.90
	HM/HMM	15.30	15.99	15.46	15.58
1M	RM	77.18	78.16	77.11	77.48
	SH	161.69	161.09	161.26	161.35
	HM/HMM	149.76	148.22	148.21	148.73

Table 2: Data Insertion Time for adding 21K Rows (in seconds). Data are collected for both with and without table index on the de-identified column.

Indexing	Method	1	2	3	Average
without Index	RM	5.33	3.51	5.39	4.74
	SH	4.93	4.85	4.54	4.77
	HM/HMM	4.37	4.40	4.36	4.38
With Index	RM	4.27	4.06	3.63	3.99
	SH	4.85	5.39	5.52	5.25
	HM/HMM	4.66	4.89	4.79	4.78

Table 3: Query time (in seconds) for fetching data for the same10K identifiers.

Method	Obs. 1	Obs. 2	Obs. 3	Average
SH	1.78	1.76	1.77	1.77
RM/HM	0.51	0.51	0.52	0.51
HMM	0.90	0.90	0.91	0.90

Table 1 shows the table creation time for initial load of 100K and 1 million records using SH, RM, and HM methods to de-identify one column.

Using similar SQL statements shown above, data insertion performance is tested with and without index on the ID column. Table 2 shows time by inserting 21K records into a table with 1 million rows using different de-identification schemes. 21K was chosen based on average daily insertions into the IW production lab test table over the past 5 years.

We choose to use a simple query that involves the join of 2 tables on index IDs to evaluate query performance.

```
SELECT count(1)
  FROM table_1, table_2
 WHERE table_1.did = table_2.did;
```

In the case of HMM, a view is used instead of an actual table. Table 3 shows query time on a query that joint select from a table with 10K identifiers and a testing table of 1 million rows over the indexed DID columns from both tables. For HMM, a view is created to replace the permanent ID with the secondary ID and query time test is performed using the view.

HM and HMM share the same table creation and insertion time, while RM and HM have similar query performance. This is due to the fact that these methods involve exactly the same database operations in the two categories, respectively.

DISCUSSION

Initial Data Loading Time

From Table 1 it is clear that data loading time is twice as fast in RM as in SH and HM/HMM. This is a cost due to the use of a secure hash algorithm. Also observed is that HM/HMM, although adding a mapping step on SH, performs better than SH. We believe that this is due to the difference between inserting a 40-byte string and a 10 digit number into the database. As with most algorithms, the database engine handles numbers more efficient than dealing with strings.

Data Insertion Time

Inserting a record does not display significant differences among the tested methods for an average daily load of 21K rows. Also noted is that there is no significant time difference in inserting into a table with or without indices that are already built on the table. This observation is in agreement with the general database practice that table index is not rebuilt or updated during insertion.

Data Update

One of the potential advantages of using HM is that DID can be refreshed using an update process. However, tests results showed that a full table update on an ID column is always much slower than a table truncate and re-build process. For example, inserting 100K records took 9.24 seconds to finish while updating 100K records on an indexed numeric DID (10% update) lasted 104.19 seconds. This process can be costly as the DDB grows possibly impairing availability of the system for customers.

HMM provides another mechanism for such an update: one simply needs to generate a new set of secondary ID's. However, as can be seen in Table 3, this process imposes a 78% penalty ((0.90-0.51)/0.51) on our query test case.

Data Deletion

Data deletion is not tested in this study because deletion rarely occurs in a de-identified data repository. For sporadic row deletions it is anticipated that this operation would not produce significant impact.

Query Performance

Since database queries rely on an index for performance, it is no surprise that a query that makes use of an index on a column of numerical values always outperforms the same query that uses an index on a column of categorical values. Therefore, it is reasonable to see that SH recorded slowest query execution time, more than 3 times that of the best performer (RM/HM) on a simple 2 table join on indexed DID columns.

The use of a view in HMM caused it to be 78% slower than direct query in RM and HM. However, this is still almost twice as fast as in SH on the same test query. It is postulated that more table joins, which is typical in data analysis, will widen the performance gap among all methods tested here.

Longitudinal Study Prevention:

In order to prevent data re-query using previous query results and to inhibit longitudinal study using the DDB, the DID used in DDB has to change periodically. With the SH method, change of the DID involves a change of the hashing function (e.g., a new "salt") and a complete rebuild of the DDB, which is impractical. The RM method requires the generation of a new set of DIDs, thus having the same limitation as in SH method. In the HM method, the random number DID, in theory, can be re-generated and the DDB updated without a complete DDB rebuild. However, in practice, the update takes a prohibitively long time, rendering it impractical. In the HMM method, an update for a new set of DID is simplified so as to re-generate the secondary random numbers. This update can be performed any time when no query is being executed in the database.

Date Shifting

Date shifting has been proposed and used in other implementations of various databases that support de-identification. For example, Roden et al reported on shifting a random number of days between 1-364 days to the past[4]. However, a date in DDB shifted in such a manner may complicate other clinical data such as diagnosis and procedure codes. Codes that have changed over time will no longer be valid for some of the patients. For example, if a patient's diagnosis code is assigned as "A" right after the new code change (from "B" to "A") takes effect, shifting this patient's visit date to the past will result in the wrong code ("B") being associated to this visit. Replacing date with a time delta from a patient's first visit will solve this problem but it is considered to be too prohibitive to some research work that relies on dates.

One possible way of solving this problem is to add a new field to the table that uses codes to store either the code descriptions or effective ending date (EED) for the code. The use of EED is possible as long as EED is not date-shifted. Date-shifted EED can obviously be used to re-calculate real patient visit date. With this mechanism, the DDB will increase in size but impacts of date shift can be minimized. Based on our query test results, the use of EED (effectively a number instead of a long string inside a database) would yield better performance but the use of such a date can be confusing when two dates in the same record show one date being shifted to the past and the other not.

Shifting date by any unknown number of days would be sufficient for the purpose of de-identification[4] alone. We tested on a shift using a patient's birthday. This is based on the observation that it is unlikely that a patient will have a hospital visit before his/her birthday. By shifting each patient's birthday to a common date, i.e., 1/1/1900, all visit dates will appear to be random while visit intervals are preserved, thus de-identified.

Patients of age over 90 at time of visit can be further time-adjusted to make his/her age 90, effectively creating an age group of 90. Patients who are less than 18 years of age at time of visit will be excluded from DDB.

Other data elements

The masking of other data elements will follow published methodologies that are HIPAA and HBP compliant[4,5]. For example, address will be modified to keep only the state and the first 2 digits of a zip code. Medical free-text reports will not be included in the DDB at this time. We feel that more evaluation is necessary on various text de-identification packages before an update to HBP can be made.

Re-identification

All schemes evaluated in this study are susceptible to re-identification attacks[11]. Clearly, published mechanisms describing re-identification attacks[14] do apply to this DDB implementation due to the inclusion of data elements such as gender, ethnicity, and clinical information such as diagnosis and procedure codes. Since the purpose of DDB is not to provide anonymized data, re-identification is not addressed in this study.

CONCLUSION

We reported here the undesirable effect of date shifting on data integrity of a DDB and suggested one possible solution to preserve the meaning of coded columns (e.g., diagnosis) against possible changes in the meaning of a code. We have also evaluated various de-identification mechanisms and their impact on different database operations.

The SH mechanism has the advantage of being relatively straight forward in design, closely preserving data model similarity between the IDB and the DDB. Moreover, this design has been deployed in production at other institutions for some time and has been

demonstrated to be effective. However, query and update performance is adversely impacted in this design as this process replaces a short integer key with a long character string that is often a join key in queries. Furthermore, more robust hashing algorithms that generate longer keys (SHA-512, for example) will provide better security but will impede query and update performance even more. The RM mechanism has the advantage of not impacting query performance when compared to the IDB as the new keys will be the same size and type as the original patient identifiers. However, DDB's produced by both SH and RM would be considered "coded" datasets by federal privacy regulations[3], leading to more confined use restrictions for research purposes.

Regenerating DID's with the HM mechanism will require an update transaction on every row of every table with an ID or will require a complete retransfer of the original IDB data into the DDB. Therefore, as data volume grows exponentially, we would expect this mechanism to be challenging to keep within a reasonable update time window. The HMM mechanism demonstrated the most practical performance when recoding the DDB since it requires update transactions per ID on only a single table in the entire data warehouse (the code map). Moreover, query performance of this model benefited from using a 10-digit integer as the key instead of a string several times that size (as in the pure hashing key approach). However, HMM suffered losses in query performance against HM due to the introduction of an extra database view even though its query performance is still advantageous over SH.

If possible, one would initially start with a DDB using HM for de-identifiers and subsequent updates using table truncate and re-load because this scheme provides the best query performance. Once the DDB grows to a size that prohibits such an update, or when demands on DDB availability render prolonged down times impractical, it can be easily converted into HMM by adding a secondary de-identifier to each identifier. This secondary de-identifier can be used to build a database view. Performance of HMM can be improved dramatically by adding a group of caching storage disks (CSD) and building materialized views (MVs) on the CSD's over the original views and queries are written using the original views. Queries to the DDB will use the MVs whenever possible but can also fall back to use the original views when MVs are not available. MVs need to be rebuilt every time secondary de-identifiers are refreshed. CSD, since it only stores the MVs and not a critical component, should be fast storage media and need not have redundancy and content backup.

REFERENCES:

1. Powell J, Buchan I. Electronic health records should support clinical research. J Med Internet Res. 2005 14;7(1):e4.
2. Langheier JM, Snyderman R. Prospective medicine: the role for genomics in personalized health planning. Pharmacogenomics. 2004, Vol. 5, 1:1-8.
3. Office for Human Research Protections (OHRP), U.S. Department of Health and Human Services. Guidance on research involving coded private information or biological specimens. October 2008.
4. Roden DM, Pulley JM, Basford MA, Bernard GR, Clayton EW, Balser JR, Masys DR. Development of a Large-Scale De-Identified DNA Biobank to Enable Personalized Medicine. Clinical Pharmacology & Therapeutics (2008); 84, 3, 362-369
5. Lyman JA, Scully K, Harrison Jr JH. The Development of Health Care Data Warehouses to Support Data Mining. Clinics in Laboratory Medicine. 2008. Vol.28, 1:55-71
6. Wylie JE, Mineau GP. Biomedical databases: protecting privacy and promoting research. Trends Biotechnol 2003;21(3):113–6.
7. de Moor GJ, Claerhout B, de Meyer F. Privacy enhancing technologies: the key to secure communication and management of clinical and genomic data. Methods Inform Med 2003;42:148–53.
8. Berman JJ. Concept-Match Medical Data Scrubbing. Archives of Pathology and Laboratory Medicine: Vol. 127, No. 6, pp. 680–686.
9. Gardner J. Xiong L. HIDE: An Integrated System for Health Information DE-identification. In 21st IEEE International Symposium on Computer-Based Medical Systems (CBMS), June, 2008.
10. Kohane IS, Dong H, Szolovits P. Health information identification and de-identification toolkit. Proc AMIA Symp. 1998; 356–360.
11. Emam KE, Jabbouri S, Sams S, Drouet Y, Power M. Evaluating Common De-Identification Heuristics for Personal Health Information. J Med Internet Res. 2006 Oct–Dec; 8(4): e28.
12. Kamal J. et al, Innovative applications of an enterprise-wide information warehouse. AMIA Annu Symp Proc. 2008 Nov 6:1134.
13. Silvey SA, Schulte J, Smaltz DH, Kamal J. Honest broker protocol streamlines research access to data while safeguarding patient privacy. AMIA Annu Symp Proc. 2008 Nov 6:1133.
14. Malin B, Sweeney L. How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems. Journal of Biomedical Informatics 2004. 37:179–192.
15. Gulcher JR, Kristjansson K, Gudbjartsson H, Stefansson K. Protection of privacy by third-party encryption in genetic research. Eur J Hum Genet 2000; 8:739–42.